Obstacle Avoiding Car

Index

Members 1
Introduction 1
Background 1
Project Specifications 1
Solution Design 2
1. Arduino Nano 2
2. L298N motor driver 3
3. HC-SR04 sensor 4
4. Sensor Holder 5
5. 360 wheel 5
6. Gear Motors 6
7. Rubber wheels 6
8. Breadboard 6
9. Jumpers 6
10. Car chassis 7
Implementation 7
Testing 12
Project Breakdown Structure 13
Results 13
Conclusion 14
Contact us 14
Work cited 14

Members

1. Saad Bin Khalid	:	20k-0161
2. M. Moaaz Bin Sajjad	:	20k-0154
3. Syed M. Daniyal Imam	:	20k-0263

Introduction:

This report is going to analyse our DLD project in detail which is an obstacle avoiding car. It is a small robot car which is able to identity obstacles and find its way safely through them without an accident.

Background:

As soon as the DLD project was announced in the class, we guys got together and exchanged ideas about different kinds of project. The project of obstacle avoiding car was stuck in my head for quite some time but I didn't disclose it immediately as I thought it would be quite difficult for us to make it. However, when I was introduced to Arduino and saw some youtube videos about it, I thought that the project is not difficult as I think it is. Hence, I discussed this with my partners and we all decided to give this project as chance.

Project Specifications:

The obstacle avoiding car consists of the following components:

No.	Component	Quantity
1	Arduino Nano	1
2	L298N motor driver	1
3	HC-SR04 sensor	3
4	Sensor holders	3
5	360 wheel	1
6	Gear Motors	2
7	Rubber Wheels	2
8	Breadboard	1
9	Jumpers	As per need
10	9V battery	2
11	Car chassis	1
12	ON / OFF switch	1

Solution Design:

Let’s have a look at brief explanation and working of each component of this project:

1. Arduino Nano:

Definition:

Arduino Nano is a small microcontroller which allows you to upload instructions in it in the form of code.

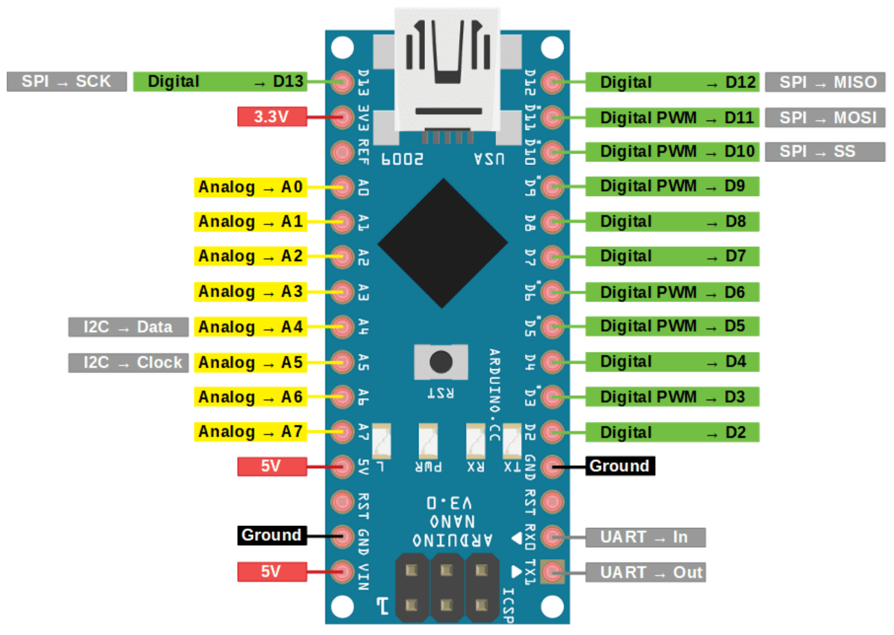
Arduino IDE:

Arduino company provides a software IDE in which you write the code for your project, then, you upload it in the microcontroller through a USB data cable. The language used in Arduino is a subset of C++ which means that you can use the basic syntax and functionalities of C++ in Arduino as well, though there are some exceptions. The Arduino provides with you powerful functions and libraries of its own which makes coding very easy for a beginner.

Reason for Choosing:

The reason for us to choose Arduino Nano as our microcontroller was that this chip is very small and light in size when compared with other variants of Arduino. Furthermore, its small size does not truncate or reduce any of its functionalities. In fact, the breadboard friendly style makes it ideal to use in small projects where efficiency cannot be comprised.

Design:



Source: Fritzing

Below is the explanation of the used pins of Arduino Nano:

- There are 13 digital pins and 8 analog pins.

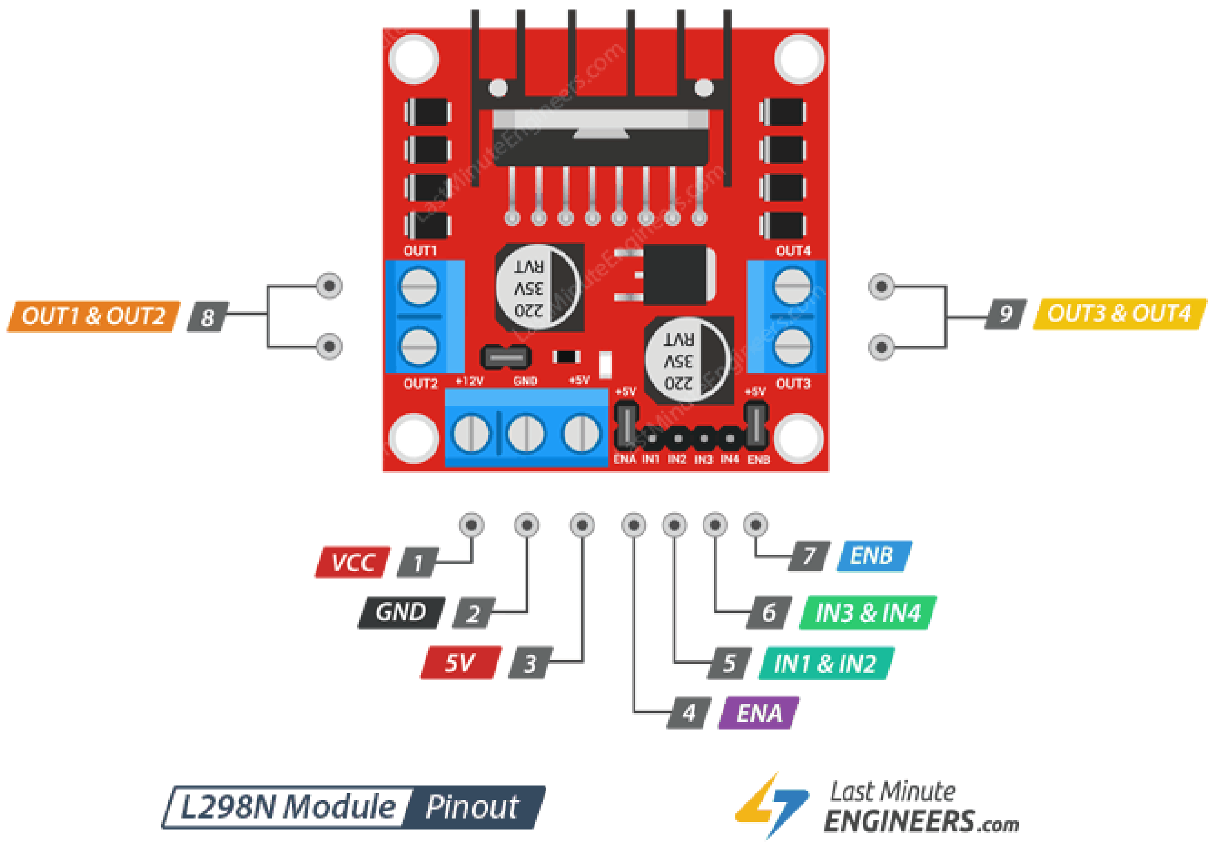
- Digital pins are used to read and write HIGH (5V) and LOW (0V) states.
- The analog pins are used to read and write analog readings which are designed in the range of 0 – 255 digits.
- Vin pin is used to power the chip. The working voltage is between 5-12V.
- The chip can also be powered through the USB data cable.
- When chip is powered through Vin, one GND must be attached with the ground of battery. The 2nd GND can be used to provide a common ground to other components.
- When chip is powered through the USB jack, both GNDs can be used to provide common grounds to other devices.
- 3.3V and 5V pins are used to provide voltages to other devices.

2. L298N motor driver:

Definition:

L298N is a relay or chip that provides the functionalities to control one or two motor(s) at the same time. One can use it to control a motor's direction and speed.

Design:



Below is the explanation of the used pins of L298N driver:

- The OUT pins are divided into two sets of 2 pins each which are used to connect motor terminals.

- VCC and GND are used to power the driver. The working voltage is 5-35V.
- There is a removeable 5V regulator exactly above the VCC and GND pins.
- The EN pins are used to read analog input in order to control speed of their respective motors.
- The two pairs of IN pins are used to control directions of their respective motor.
- If IN1 and IN2 at the same level, motor 1 will be stopped. If they are at different levels, motor may spin clockwise or anticlockwise. If the levels are reversed, motor will spin in the opposite direction. Same is the case for IN3 and IN4.

3. HC-SR04 sensor:

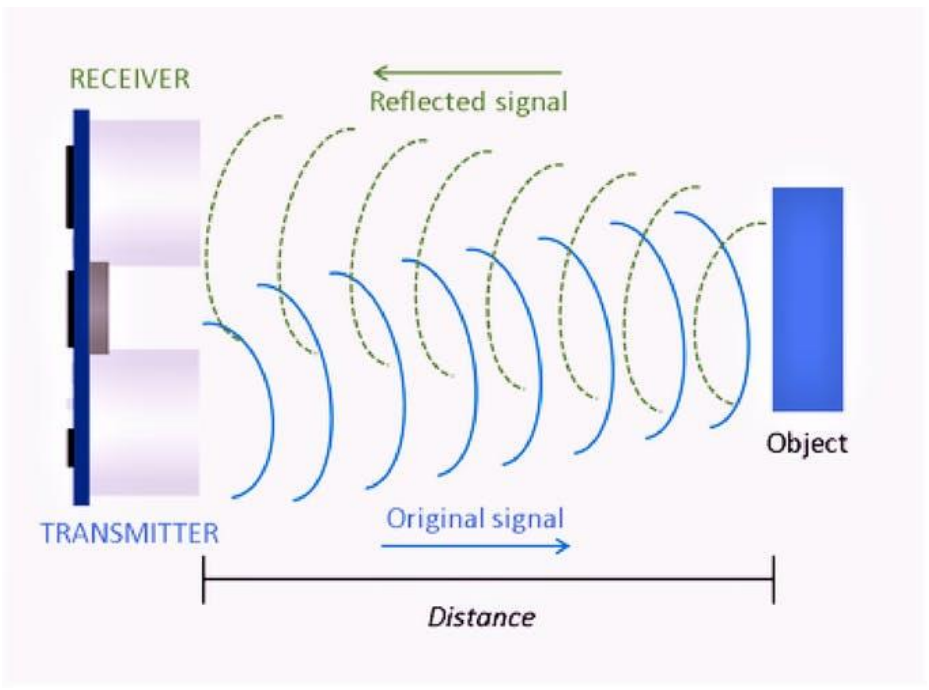
Definition:

HC-SR04 sensor is a device which uses its ultrasonic signals to locate any object in its vicinity.

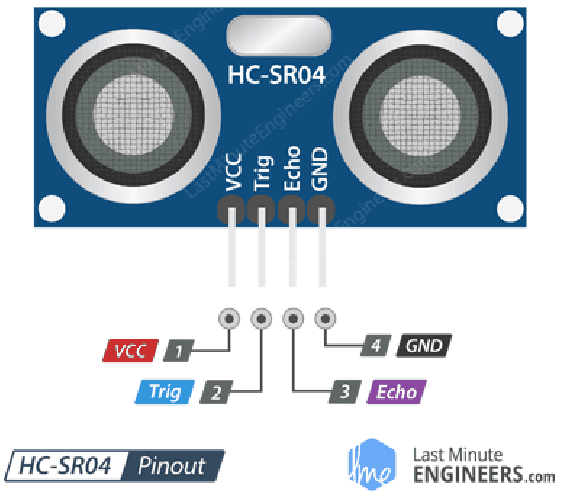
Working:

The working of this device is quite similar to the working of a radar. The sensor consists of two parts: Emitter and Receiver.

- Emitter propagates ultrasonic signals in its direction.
- Receiver is responsible to detect the reflected ultrasonic signals.
- When a reflected signal is received, the device echoes back a signal.



Design:



- The VCC is pin is used to power the sensor. The input voltage is 5V.
- The GND is used to provide ground to the sensor.
- The TRIG pin is used to trigger the pulse signal. It should be HIGH for a moment and then LOW again in order to generate a pulse.
- The ECHO pin is used to receive the signal. The pin is read from Arduino or any other microcontroller to decode the signal in order to retrieve the elapsed time.

4. Sensor Holder:

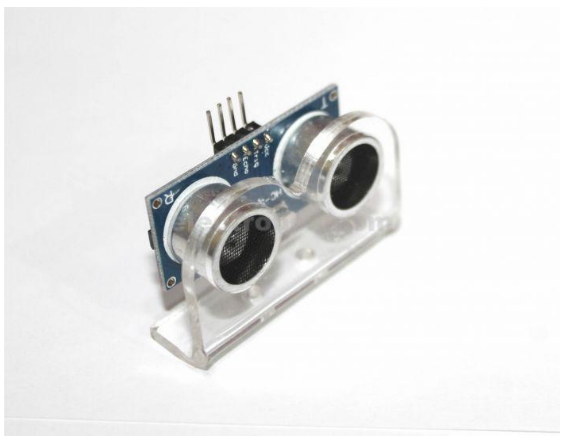
Definition:

A holder which is used to safely mount the HC-SR04 or similar sensors.

5. 360 wheel:

Definition:

The wheel which is able to move and rotate in any direction of applied pressure.



6. Gear Motors:

Definition:

These are powerful DC motors which are used to rotate the wheels of car.

Working:

- A voltage in the range of 3-12V must be applied at the terminals of motor in order to rotate it.
- The rate of rotation is proportional to the amount of applied voltage.
- Reversing the polarity of the voltage changes the direction of rotation.

7. Rubber wheels

Definition:

These are toy wheels which are made up of rubber and have a very strong grip on most of the surfaces.

8. Breadboard:

Definition:

Breadboard is a plastic board with a lot of connection terminals in order to provide re-useable, easy, and safe connections.

9. Jumpers:

Definition:

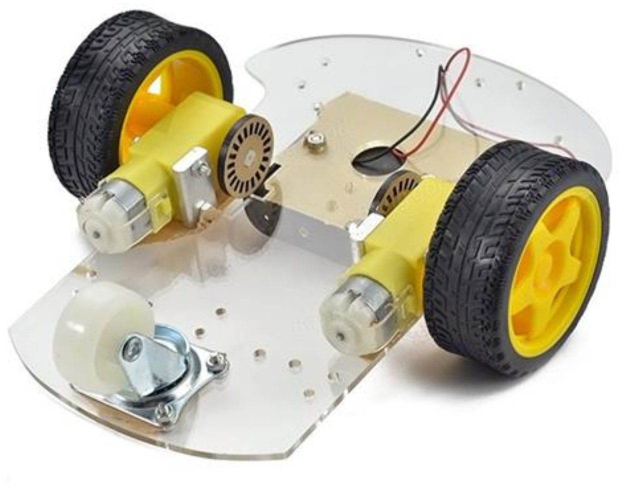
These are connecting wires which are used to connect components of a circuit. The kinds of jumpers used in our project are:

- Male to Male jumpers
- Male to Female jumpers

10. Car chassis:

Definition:

The chassis is the base of a toy car which is used to hold the body of our project. It includes a number of screw holes and space to insert other components like: motors, switches, sensors etc.



Implementation:

Now that we know the function and details of each component, let's explore the implementation of the project by combining all of them together.

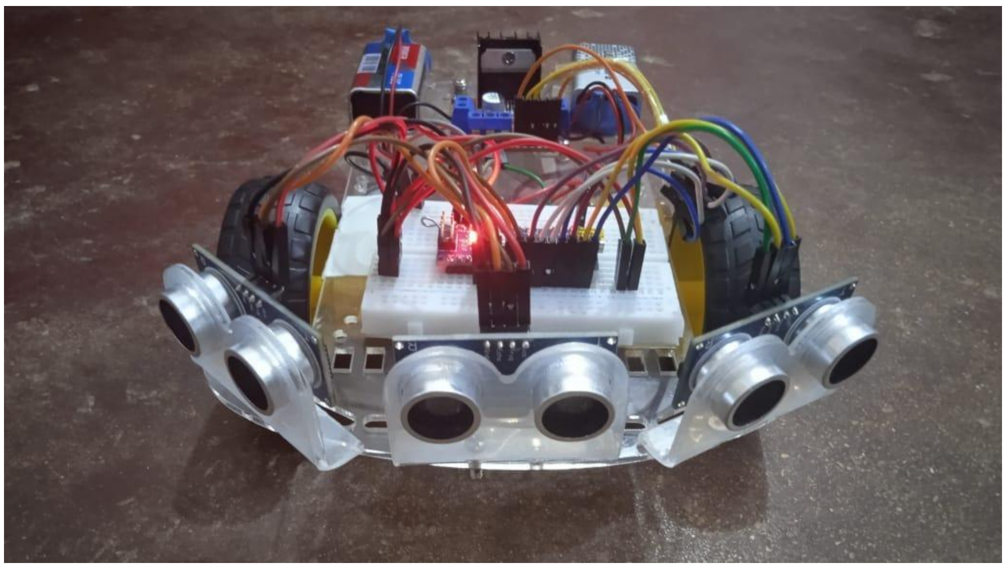
Structure:

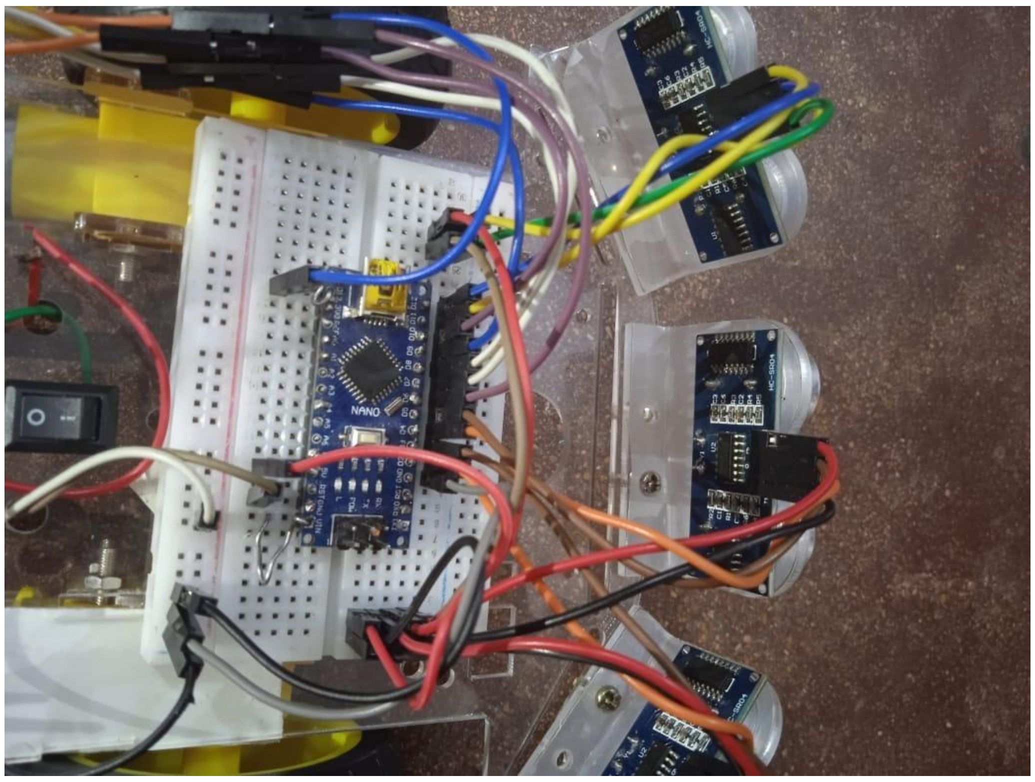
- The tyres are attached with the gear motors.
- The motors are attached with the car chassis.
- The L289N motor driver is also screwed down with the chassis.
- The breadboard is taped to the chassis with a double-sided tape.
- Two 9V batteries are also taped to the chassis.
- The Arduino is mounted on the breadboard.
- The sensors are mounted on the sensor holders which are screwed down at the front side of the chassis.
- A switch is also attached with the chassis.

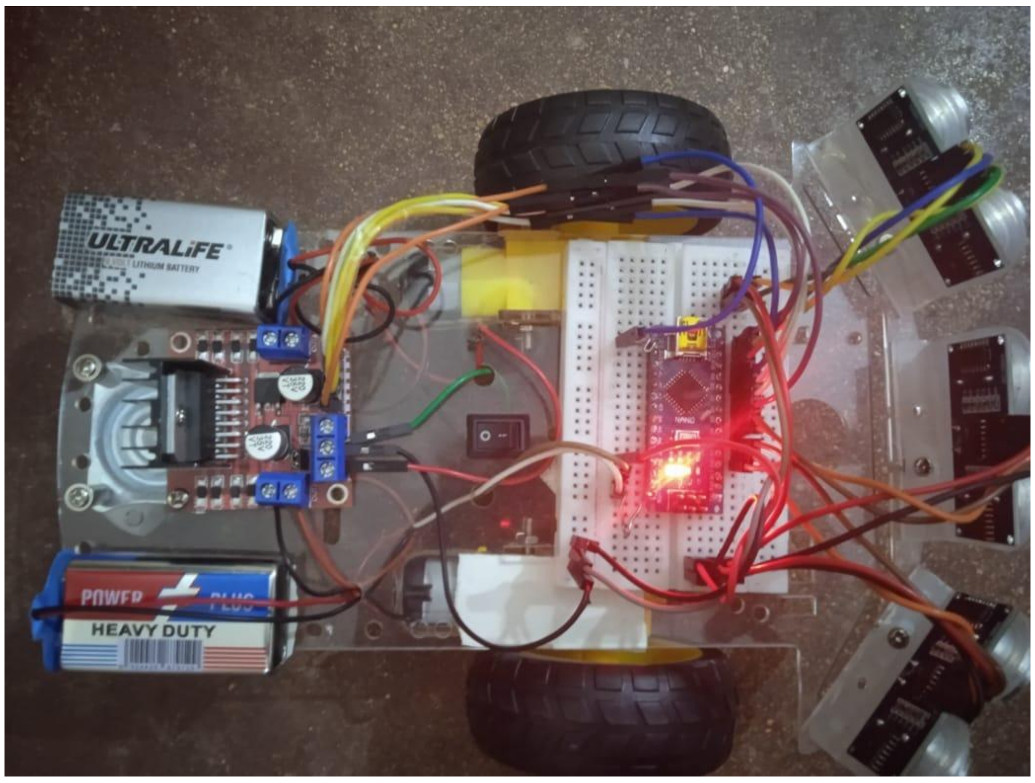
Connections:

- One battery is used to power the Arduino while the other is used to power the motor driver.
- The battery which is used to power motor driver has to go through a two-state switch.
- The motors' terminals are connected with the L298N driver.
- The necessary pins of L298N (explained above) are connected with the pins of Arduino nano.
- The 5V ang GND pins of Arduino are used to power the sensors.
- The remaining pins of sensors are also connected with arduino.

Screenshots:







Implementation:

The working of the car is mentioned in the following points:

- First of all, the code needs to be uploaded to the chip through a micro-USB data cable.
- The Arduino is powered by the 9V battery.
- The switch is turned on to power the driver.
- As soon as the motor is turned on, tyres will start moving at the specified speed.
- The three sensors give feedback to the Arduino which uses these signals to decide where and when to turn the car.

Movement of tyres:

- When the car has to turn left, right tyre is stopped and left tyre is rotated in forward direction.
    - Forward direction mean IN1 is HIGH and IN2 is LOW
    - Stop state infers that IN2 and IN3 are at same level.
- Similarly, when the car has to turn right, left tyre is stopped and right tyre is rotated in forward direction.
    - Stop state infers that IN1 and IN2 are at same level.
    - Forward direction means IN3 is HIGH and IN4is LOW
- When car has to move forward, both tyres are rotated in forward direction.
    - Forward direction means IN1 and IN3 are HIGH while IN2 and IN4 are LOW.
- Similarly, when car has to move back, both tyres are rotated in reverse direction.
    - Reverse direction means IN1 and IN3 are LOW while IN2 and IN4 are HIGHs.
- The speed of the car is decided in the code and remains constant throughout the operation.

Sensors' feedback:

Every sensor is designed to send a feedback to Arduino whenever any object reflects back the emitted ultrasonic waves.

- The sensors are continuously being triggered by a pulse through the TRIG pin. Then, the signals are also continuously being read by the Arduino through ECHO pins.
- The design of generating the pulse is specified in the code.
- The process of reading is also specified in the code.

Code:

We feel it important to mention that the code for this project is NOT COPIED from the internet. Even-though there were a number of *sources available on the internet, the developer of this code decided to write his own program and libraries.*

The code consists of following files:

- motor.ino - the main program.
- Wheel.h – contains the class to control a wheel.
- Car.h – contains the class to control a car.

We are going to look at the motor.ino only as other files are programmed for general purposes. However, the examiner of this report is free to ask anything about those files as well.

motor.ino:

This program is the running program for our project. It contains objects of other user-defined classes, declared variables and functions.

```
#include "Car.h"

Car myCar(150); //  speed of the car
//  in inches :
const double safe = 10, danger = 3, minSafe = 6;
double left, right, mid;

//  makes sure that the car is stays within the
//  safe distance
void adjust()
{
   if (mid <= danger)
   {
      while (mid <= minSafe)
      {
         myCar.back();
         mid = myCar.SignalMid();
      }
   }

   if (left <= danger || right <= danger)
   {
      while (left <= minSafe || right <= minSafe)
      {
         myCar.back();
         right = myCar.SignalRight ();
         left = myCar.SignalLeft ();
      }
   }
}

void setup()
{
// nothing
}

```

```
void loop()
{
    left  = myCar.SignalLeft();
    right = myCar.SignalRight();
    mid   = myCar.SignalMid();

    // if there is nothing in the way
    if (left > safe && mid > safe && right > safe)
    {
        myCar.forward();
    }

    // if right detects something
    if (left > safe && mid > safe && right <= safe)
    {
        adjust();
        myCar.left();
    }

    // if mid detects something
    if (left > safe && mid <= safe && right > safe)
    {
        adjust();

        if (right < left)
            myCar.left();
        else
            myCar.right();
    }

    // if left is open only
    if (left > safe && mid <= safe && right <= safe)
    {
        adjust();
        myCar.left();
    }

    // if left detects something
    if (left <= safe && mid > safe && right > safe)
    {
        adjust();
        myCar.right();
    }

    // if mid is open only
    if (left <= safe && mid > safe && right <= safe)
    {
        adjust();

        if (right < left)
            myCar.left();
        else
            myCar.right();
    }
    // if right is open only
    if (left <= safe && mid <= safe && right > safe)
    {
        adjust();
        myCar.right();
    }

    // if the way is blocked completely
    if (left <= safe && mid <= safe && right <= safe)
    {
        adjust();
      }
}
```

Variables:

- The “`left`”, “`mid`”, and “`right`” variables are used to store the distances in inches.
- The “`safe`” stores the maximum safe distance in inches. It is the distance in which car is allowed to make a turn freely.
- The “`danger`” stores the dangerous distance in inches. It is the distance in which car is not allowed to proceed any further.
- The “`minSafe`” stores the shortest safe distance i.e. the distance the car should always maintain. When the car is in “`danger`” distance, it will return back to the “`minSafe`”.

Functions:

- The “`adjust()`” function is a very important function which makes sure that the car is always in the “`minSafe`” range. That’s why we have used it with every condition in the continued code.
  Car.h:
- The “`SignalLeft()`”, “`SignalRight()`”, and “`SignalMid()`” member functions of class `Car` trigger the left, right, and middle sensor respectively. They also receive the signal back from their respective sensors and return the calculated distance in inches.
- The “`forward()`”, “`back()`”, “`left()`”, and “`right()`” moves the car forward, back, left, and right respectively.

Testing:

Now that we know the working and implementation of the car, let us have a look at the tests that were performed on this car in order to make sure that there is nothing wrong with it.

Click here to have a look at the testing video

The video contains following tests:

1) Wide maze test – the car is surrounded by wide walls; however, an opening is provided for the car to find its way out.
2) Corner test – the car is released while facing a corner.
3) Small maze test – the car is surrounded by narrow walls; however, an opening is provided for the car to find its way out.

We are glad to announce that the car successfully passed all of these tests.

Fallback:

There is actually a kind of loophole in this project which is related to the working of sensors. Since the sensors emits ultrasonic sound waves, hard and stiff objects reflect them back very well and the results are pretty smooth. However, when the car encounters soft objects, the sound waves are not reflected back properly, and the car assumes that there is no obstacle at all. Due to this, the car is known to bump into soft objects like pillow, curtains etc.

Therefore, we feel that the reader should be aware of this disadvantage.

Project Breakdown Structure:

We divided the project structure into three major parts:

- Coding
- Connecting components
- Designing the car's structure.

Saad (20k-0161) handled the coding, Moaaz (20k-0154) was responsible for all connections & components, and Daniyal (20k-0273) was responsible for the basic structure and design of the car.

Timeline:

The designing and basic structure of the car was dealt first. Then, all connections and components were attached in their right place. Finally, the code was developed and uploaded in Arduino nano.

Results:

All of this hardwork and struggle results in a very reliable and efficient car which is able to avoid the objects automatically. Despite its impressive results, there are some important points which should not be forgotten:

1. The user should keep in mind that the Arduino chip must not be overpowered or else it might get burned. The voltage range for Arduino is 5-12V.

2. The motor driver voltage range is 5-35V.
3. When the motor driver is powered more than 12V, make sure to remove the 5V regulator from its place or else the motor diver will get burnt out.
4. The sensors should also not be powered more than 5 volts.
5. The Arduino must not be powered from both its VCC pin and the USB jack.
6. All of the components of the circuit must share a common ground or else the circuit will not run.

Conclusion:

To conclude our report, we came all the way from scattered components to a well arranged and working car. The brain of the car was programmed through Arduino, the objects were detected by sensors, the motors were controlled by motor driver, and tyres with strong grips were selected in order to provide a smooth turnover.

It was not easy to make this project in Lockdown, especially in Ramadhan, but our team still worked hard in this project. In the end, it was worth it as we learned and were exposed to completely different and interesting things.

We hope that you like our project, and we always look forward to such interesting projects in the future.

Contact us:

1. Saad Bin Khalid – k200161@nu.edu.pk
2. M. Moaaz Bin Sajjad – k200154@nu.edu.pk
3. Syed M. Daniyal Imam – k200273@nu.edu.pk

Work cited:

1. **Arduino Playlist - *Paul McWhorter:*** Youtube link.
2. **Arduino's Doc – *Arduino Company:*** Link to website.